```
=> d his

     (FILE 'USPAT' ENTERED AT 13:05:43 ON 09 JUL 1998)
L1          1569 S (PRIORIT? (3A) (QUEUE# OR BUFFER#))
L2           294 S L1 (P) (UNDERRUN OR OVERRUN OR FULL#### OR EMPTY)
L3           125 S L1 (7A) (UNDERRUN OR OVERRUN OR FULL#### OR EMPTY)
L4            32 S L3 (P) STOR?
L5           470 S (HIGH## OR LOW##) (5A) L1
L6           144 S (HIGH## OR LOW##) (P) L2
L7            66 S (HIGH## OR LOW##) (P) L3
L8            11 S L7 (P) STOR?
L9           229 S L1 (7A) STOR?
L10            1 S L9 (P) (OVERRUN OR UNDERRUN OR (SPACE (2A) AVAILABL?))
L11           97 S L1/AB
L12            7 S L9/AB
```

US PAT NO: 5,724,358 [IMAGE AVAILABLE] L12: 1 of 7
TITLE: High speed packet-switched digital switch and method

ABSTRACT:
A . . . and priority level of the data packet. Pointers to buffer locations containing data packets having a particular priority level are **stored** in one or more **priority** sub-**queues** for one or more of the plurality of output ports based on the output port destinations and the priority level. . .

US PAT NO: 5,678,026 [IMAGE AVAILABLE] L12: 2 of 7
TITLE: Multi-processor data processing system with control for granting multiple storage locks in parallel and parallel lock priority and second level cache priority queues

ABSTRACT:
A . . . an address, even if the address for which the lock was requested is not local relative to the processor. Parallel **priority queues** are employed for parallel handling of **storage** lock functions and general storage operations, thereby reducing contention for priority between storage lock operations and general storage operations where. . .

US PAT NO: 5,630,123 [IMAGE AVAILABLE] L12: 3 of 7
TITLE: Software system utilizing a filtered priority queue and method of operation

ABSTRACT:
A . . . 22, 24, 26 and 28) of a priority queue and to filter and arrange the data records in a memory **storage** device (8) to form the filtered **priority queue** (10). The filtered **priority** queue (10) comprises a remaining set (12) and a filtered set (14). The filtered set (14) contains a first subset. . .

US PAT NO: 5,521,916 [IMAGE AVAILABLE] L12: 4 of 7
TITLE: Implementation of selective pushout for space priorities in a shared memory asynchronous transfer mode switch

ABSTRACT:
A . . . packets are stored in these priority subqueues using a first pointer pointing to the next packet of the same space **priority** in the **queue**. The second pointer associated with a **stored** packet points to the previous packet of greater than or equal space priority in the FIFO order in the queue.. . . to the next packet in FIFO order in the queue, and this field is used by a processor to decide **priority** sub-**queues** to serve next. The packets are **stored** in the queues in a FIFO order using the two pointers and the fields of the packets. The processor controls. . .

US PAT NO: 5,150,358 [IMAGE AVAILABLE] L12: 5 of 7
TITLE: Serving constant bit rate traffic in a broadband data switch

ABSTRACT:
This . . . Each member of the group for queuing data of a different band of bit rates. Data is transmitted from these **queues** with highest **priority** from the **queue storing** data of the highest band of

bit rates. If the longest enqueued entity of data in one of the other. . .

US PAT NO: 5,043,981 [IMAGE AVAILABLE] L12: 6 of 7
TITLE: Method of and system for transferring multiple priority queues into multiple logical FIFOs using a single physical FIFO

ABSTRACT:
An . . . of priority. The network on which the FDDI is implemented includes a plurality of processors each having a system for **storing** the frames of data in **queues** corresponding to **priority**, and an output buffer configured to have a plurality of logical FIFOs corresponding to the queues. Data is transferred one. . .

US PAT NO: 4,630,261 [IMAGE AVAILABLE] L12: 7 of 7
TITLE: Integrated buffer management and signaling technique

ABSTRACT:
In . . . output time map and a controller. The controller tags incoming data (be it a normal message and/or signaling information) and **stores** the data into the **priority buffer**. The tag may include a priority classification, identification, entry-time, etc. Signaling information is assigned a priority I designation while newly. . . .

US PAT NO: 5,130,983 [IMAGE AVAILABLE] L8: 10 of 11
TITLE: Method of polling to determine service needs and the like

DETDESC:

DETD(59)

To implement priority based message handling using the **high** order bits as with DAT, it is best to maintain multiple queue arrays QJ an QX, one set for each priority. Let us assume for a moment we want to provide two message priorities, **high** and **low**. **High** priority messages come from addresses beginning with 4 through 7, **low** from addresses beginning with 0 through 3. When a polling sequence is processing the **high** priority messages, a complete sequence as described above occurs. However, after processing a **lower** priority polling cycle 220, the controller must immediately issue a level 0 poll to see if any **high** priority messages are to be processed. This special polling cycle is processed in steps 273 of FIG. 18. If there are **higher** priority messages, the controller then starts processing a **high priority queue** until **empty**. The terminals with **low** priority messages, that had their addresses partially satisfied, wait in step 273 until polling continues back at their **lower** priority level, and then resumes at step 274. This process is facilitated by adding message priority to the poll frame 75. When a **higher** priority queue is exhausted in step 243, the controller switches to a **lower** priority queue and repeats step 243. This process can also be facilitated by using another stack array QP to **store** the message priority, and maintaining the stack in order by priority before last in first out (LIFO). If priority enforcement. . .

US PAT NO: 5,123,045 [IMAGE AVAILABLE] L8: 11 of 11
TITLE: Comprehensive software protection system

CLAIMS:

CLMS(15)

15. . . . method of hiding from an observer a pattern of access to memory by a program, comprising the steps of:

a) **storing** the program and the data the program uses, comprised of a plurality of virtual memory locations specified by virtual addresses,. . . held in the memory wherein a physical address of a physical memory location in which a virtual memory location is **stored** is specified by a pseudo-random function of its virtual address;

b) accessing each buffer whenever a memory access is sought;

c) when. . . is located by the program, moving contents of the location to a lowest level buffer; and

d) when a buffer is **full**, moving its contents to a next **higher priority buffer** and pseudo-randomly rearranging the sequence in which the virtual memory locations are held in the next **higher** priority buffer.